

Chapter 12

Software Tools for Optimization Processes

Optimization problem developed for different processes are very complicated. Mostly, the number of variables and constraints are very large. These complicated problems are difficult to solve by hand. Therefore, we take help of computer to solve those problems with very less effort that save our time and money. Computer programming solves these optimization problems using some numerical techniques. The algorithms are discussed in previous chapters. Some commercial software/software tools are available that can be used for optimization purpose. The most popular software/software tools are LINGO, MATLAB, MINITAB®, GAMS etc. In this chapter, we will discuss these software/software tools with some examples.

12.1 LINGO

Introduction to LINGO

LINGO is modeling language and optimizer developed by LINDO Systems Inc. It is a simple tool for formulating large problems concisely, solve them, and analyze the solution using the power of linear and nonlinear optimization.

The outer window shown in Fig. 12.1, labeled **Lingo 14.0**, is the LINGO main frame window. All other sub-windows will be enclosed within this main frame window. The entire command menu and the command toolbar are placed at the top of the frame window. The status bar at the lower edge of the main frame window gives us various information related to the current state of LINGO. Both the status bar and the toolbar can be suppressed using the *LINGO|Options* command. The main window consists of a sub-window labeled **Lingo Model-Lingo1** is used for Lingo Model. Here we will demonstrate how to enter and solve a small model in Windows platform. The text of the model's equations identical on all platforms, it is platform independent. Though, the procedure for entering a model is somewhat different on non-Windows platforms.

When we start LINGO in Windows platform, screen should look like the following:

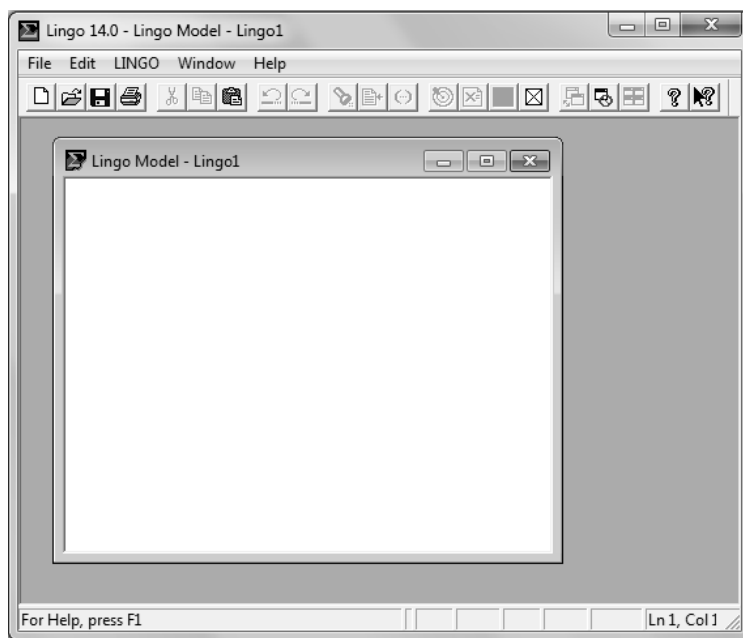


Fig. 12.1 LINGO main window

Entering the Model

Example 12.1

We will now solve the problem 6.1 using LINGO.

The problem formulated in chapter 6 is given below:

$$\text{Max } F = 14x_{p_1} + 11x_{p_2} \quad (6.12a)$$

subject to

$$3x_{p_1} + 2x_{p_2} \leq 36 \quad (6.12b)$$

$$x_{p_1} + x_{p_2} \leq 14 \quad (6.12c)$$

$$x_{p_1}, x_{p_2} \geq 0 \quad (6.12d)$$

After entering the above model within the **Lingo Model-Lingo1** window, our model window should look like this:

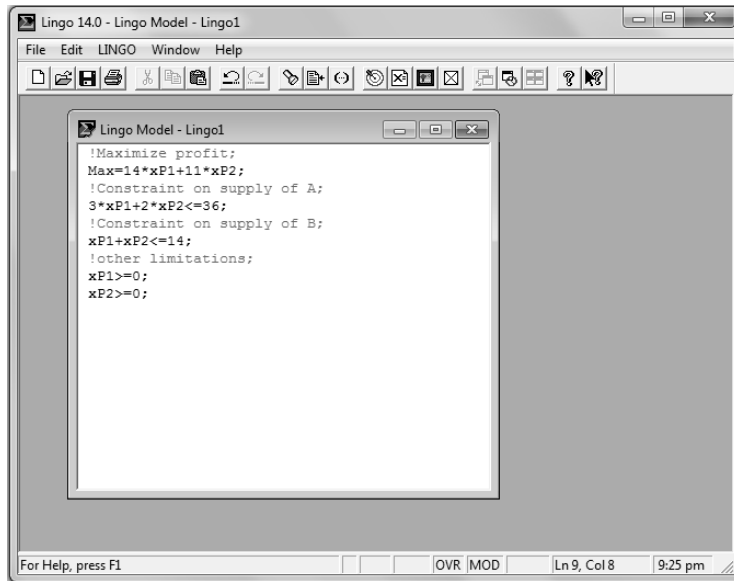


Fig. 12.2 Lingo Model-Lingo1 window with a maximization model

Solving a LINGO Model


After entering the LINGO model into the LINGO Model window, the model can be solved by clicking the *Solve* button on the toolbar, by choosing LINGO|Solve from the menus, by using the keyboard shortcut “ctrl + U” or by pressing the  button from toolbar menu as shown below



Fig. 12.3 Toolbar of LINGO

If there is any syntax error, you will get an error message as given below:

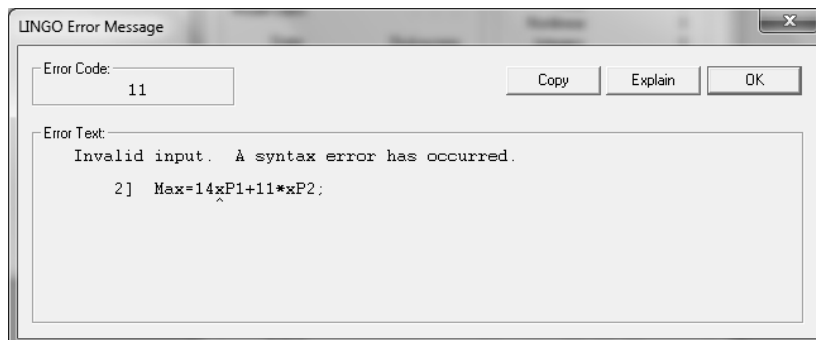


Fig. 12.4 LINGO error message box

Solver Status Window

If there are no syntax errors, LINGO will call up the suitable internal solver to start searching for the optimal solution to our model. When the solver starts working, a *solver status* window appears on the screen as shown in Fig. 12.5:

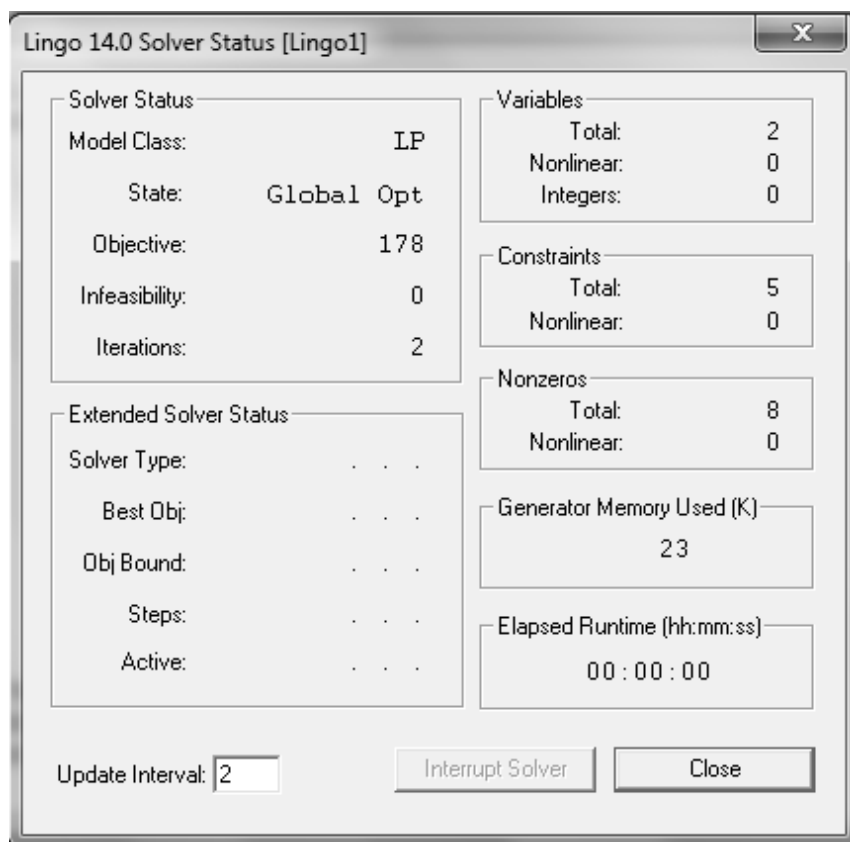


Fig. 12.5 Solver status window

This solver status window gives us information on the number of nonlinear, integer, and total variables in our model; the nonlinear and the total number of constraints used in the model; and the number of nonlinear and total nonzero variable coefficients used. The details of the model classification like LP, QP, ILP, IQP, NLP, etc. is given in the Solver Status box within this window. This also provides the status of the current solution (local or global optimum, feasible or infeasible, etc.), the value of the objective function, the infeasibility of the model (amount constraints are violated by), and the number of iterations required to solve the model. Similar information is available in the Extended Solver Status box for more advanced branch-and-bound, global, and multistart solvers.

LINGO solution report window (Fig. 12.6) gives many parameters like reduced cost, dual price, slack or surplus etc.

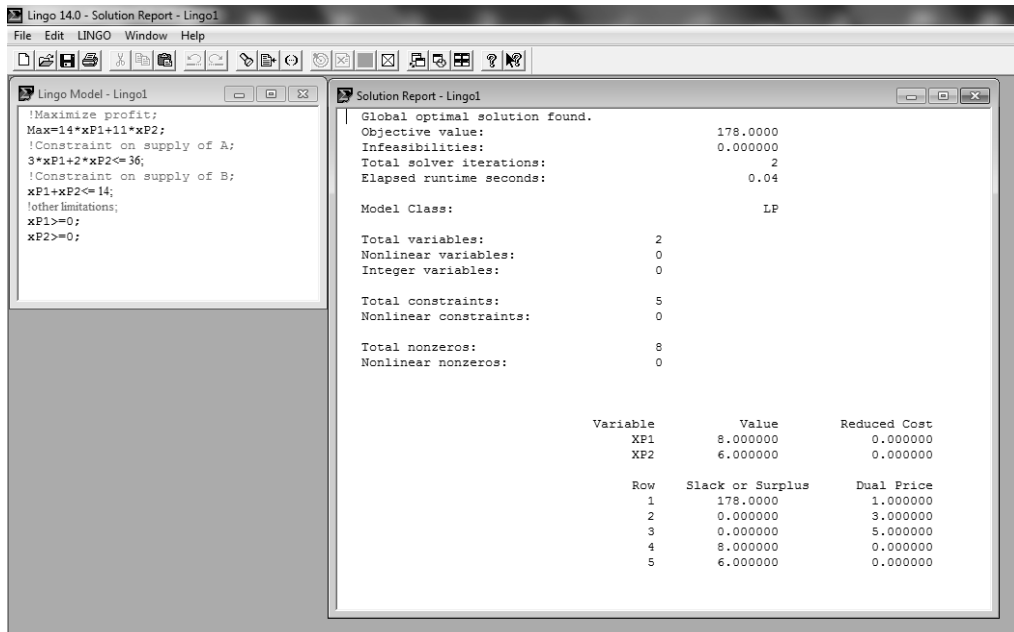


Fig. 12.6 LINGO solution report window

For variables, which are not included in the optimal solution, the reduced cost gives us an idea about how much the objective function value would decrease (for maximization) or increase (for minimization) if one unit of that variable were to be included in the solution. For instance, if for a certain variable the reduced cost 4, then the optimal value of the maximization problem would decrease by 4 units if 1 unit of that variable is to be added. The value of the reduced cost for any variable, which is included in the optimal solution is always zero. We can get an idea about how tight the constraint is, from the Slack or Surplus column in the Solution Report window. The value of slack or surplus is zero when a constraint is completely satisfied as an equality. The positive of slack/surplus indicates how much of the variable could be added to the optimal solution to make the constraint equality. When the slack/surplus is negative, the constraint has been neglected. The Dual Price column explains the improvement in the objective function if the constraint is relaxed by one unit.

General LINGO Syntax

- An expression in LINGO may be broken up into several lines as we want, but the expression must be completed with a semicolon. In the example, we have been used two lines rather than just one to write the objective function:
 Max: 14*xP1
 + 11*xP2;
- We can also enter some comments for better understanding of the readers (Fig. 12.2). Comments start with an exclamation point (!) and terminate with a semicolon (;). When LINGO solves the model, it overlooks all the text between an exclamation point and terminating semicolon.

Comments can be written in more than one line and can share lines with other expressions in LINGO. For example:

Max: 14*xP1 !Profit from product P1; + 11*xP2; !Profit
from product P2;

- iii. In LINGO, the uppercase and lowercase in variable names have the same meaning they are not distinguishable. Therefore, the variable names given below would all be considered equivalent:

PROFIT

Profit

profit

When creating variable names in LINGO, all names must start with an alphabetic character (A–Z). Subsequent characters may be either alphabetic, numeric (0–9), or the underscore (_). The length of the variable names may be up to 64 characters.

Using Sets in LINGO

LINGO allows us to group many occurrences of the same variable into sets. When the number of variables is large, we can represent it as a set. Sets are either primitive or derived. A primitive set is one that consists of distinct members only. Whereas a derived set, contains other sets as its members.

Example 12.2

The warehouse problem can be written easily using sets in LINGO.

Consider, a Wireless Widget (WW) Company has five warehouses supplying their widgets to six vendors. Each warehouse has a limit of widgets supply that cannot be exceeded, and each vendor has a demand for widgets that must be fulfilled. The company wants to decide how many widgets to distribute from each warehouse to each vendor to minimize the total transportation cost.

Solution

This problem can be represented as follows:

$$\text{Minimize } f = \sum_{i=1}^5 \sum_{j=1}^6 c_{ij} x_{ij} \quad (12.1a)$$

$$\text{subject to } \sum_{i=1}^5 \sum_{j=1}^6 x_{ij} \leq b_j \quad (12.1b)$$

$$\sum_{i=1}^5 \sum_{j=1}^6 x_{ij} \leq d_i \quad (12.1c)$$

$$x_{ij} \geq 0, \quad (i = 1, 2, \dots, 5; j = 1, 2, \dots, 6) \quad (12.1d)$$

The corresponding dual problem can be given by

The values of the c_{ij} , d_i , and b_j are given in Table 12.1, 12.2, and 12.3

Table 12.1 Widget capacity data

| Warehouse | Widget on hand |
|-----------|----------------|
| 1 | 60 |
| 2 | 55 |
| 3 | 51 |
| 4 | 41 |
| 5 | 52 |

Table 12.2 Vendor widget demand

| Vendor | Widget demand |
|--------|---------------|
| 1 | 35 |
| 2 | 37 |
| 3 | 22 |
| 4 | 43 |
| 5 | 38 |
| 6 | 32 |

Table 12.3 Shipping cost per widget (\$)

| | V1 | V2 | V3 | V4 | V5 | V6 |
|-----|----|----|----|----|----|----|
| WW1 | 6 | 2 | 7 | 4 | 2 | 9 |
| WW2 | 4 | 9 | 5 | 3 | 8 | 6 |
| WW3 | 7 | 6 | 7 | 3 | 9 | 2 |
| WW4 | 2 | 3 | 9 | 7 | 5 | 3 |
| WW5 | 5 | 6 | 2 | 3 | 8 | 1 |

The corresponding LINGO code:

```

MODEL:
! A Transportation Problem with 5 Warehouses and 6 Vendors;
SETS:
WAREHOUSES: CAPACITY;
VENDORS: DEMAND;

```

```

LINKS( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
! Data is given here;
DATA:
!set members;
WAREHOUSES = WH1 WH2 WH3 WH4 WH5;
VENDORS = V1 V2 V3 V4 V5 V6;
!attribute values;
CAPACITY = 60 55 51 41 52;
DEMAND = 35 37 22 43 38 32;
COST = 6 2 7 4 2 9
4 9 5 3 8 6
7 6 7 3 9 2
2 3 9 7 5 3
5 6 2 3 8 1;
ENDDATA
! The objective function;
MIN = @SUM( LINKS( I, J):
COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( VENDORS( J):
@SUM( WAREHOUSES( I): VOLUME( I, J)) =
DEMAND( J));
! The capacity constraints;
@FOR( WAREHOUSES( I):
@SUM( VENDORS( J): VOLUME( I, J)) <=
CAPACITY( I));
END

```

Variable Types in LINGO

All LINGO variables model are considered to be continuous and non-negative unless otherwise specified. The default domain for given variables can be override by using LINGO's four variable domain functions. These variable domain functions are given below:

- i. @FREE – any positive or negative real value
- ii. @GIN – any positive integer value
- iii. @BIN – a binary value (i.e., 0 or 1)
- iv. @BND – any value within the specified bounds

Similar syntax is used for the @GIN, @FREE, and @BIN variable domain functions. The general form for the declaration of a variable x using any of these functions is @FUNCTION(X); (see example 12.3)

The @BND function has a somewhat different syntax, which consists of the lower and upper bounds for the acceptable variable values. We can declare a variable x between a lower bound and an upper bound is given by @BND(lower bound, X , upper bound).

Example 12.3

Solve the problem given in Example 6.6.

Solution

Maximize: $Z = 5x_1 + 7x_2$

subject to constraints

$$x_1 + x_2 \leq 6$$

$$5x_1 + 9x_2 \leq 43$$

$$x_1, x_2 \geq 0 \text{ and Integer}$$

The given integer programming problem can be solved using LINGO with the simple code

```
!Code for Example 6.6;
max=5*x1+7*x2;
x1+x2<=6;
5*x1+9*x2<=43;
x1>=0;
x2>=0;
@GIN(x1);
@GIN(x2);
```

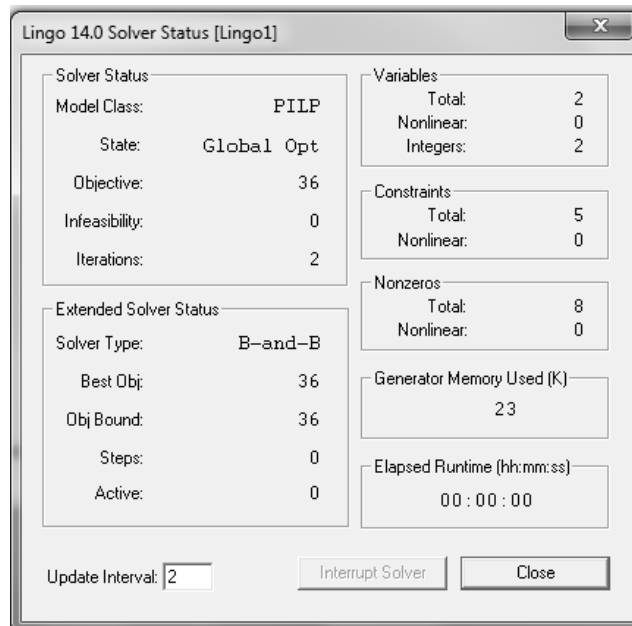


Fig. 12.7 LINGO solver status window

The model class PILP represents Pure Integer Linear Programming where all variables are restricted to integer values and all expressions are linear.

The optimum solution is $Z_{\max} = 36$; $x_{1\max} = 3$, $x_{2\max} = 3$.

Modeling from the Command-Line

Following the steps:

LINGO>Window>Command Window

The LINGO command window will appear. The colon character (:) at the beginning of the screen is LINGO's prompt for input. When we see the colon prompt, it indicates that LINGO is expecting a command. When we see the question mark prompt, we have already started a command and LINGO is asking us to supply additional information associated to this command such as a name or a number.

Note All screenshot are taken with permission from Lindo Systems Inc.

12.2 MATLAB

There are three main toolboxes which are generally used for optimization purposes.

i. Curve fitting toolbox

Graphical tools and command-line functions for fitting curves and surfaces from data are available in Curve Fitting Toolbox™. We can perform exploratory data analysis using this toolbox. Pre-processing and post-processing of data, comparing candidate models, and removing outliers are possible using this toolbox. The inbuilt library of linear and nonlinear models can be used to conduct regression analysis. We can also specify our customized model equations. The library provides us the starting conditions and optimized solver parameters to enhance the quality of our results. The nonparametric modeling techniques like splines, interpolation, and smoothing are also possible with this toolbox.

ii. Global optimization toolbox

Global Optimization Toolbox provides techniques, which search for global solutions to problems that have multiple stationary points. This toolbox consists of solvers like global search, multistart, pattern search, genetic algorithm, and simulated annealing. We can employ these solvers to optimize problems where the objective or constraint functions are continuous, discontinuous, stochastic, or even does not possess derivatives.

Pattern search and genetic algorithm solvers support algorithmic customization. We can develop a customized genetic algorithm variant by changing the initial population and fitness scaling options. Parent selection, crossover, and mutation functions can also be defined as per our requirement. Customization of the pattern search is done by defining polling, searching, and other functions.

iii. Optimization toolbox

Optimization Toolbox™ comprises algorithms for standard and large-scale optimization. These algorithms solve unconstrained and constrained continuous and discrete problems.

This toolbox comprises functions for linear programming, quadratic programming, nonlinear optimization, nonlinear least squares, solving systems of nonlinear equations, binary integer programming, and multi-objective optimization.

Table 12.4 shows different functions used for optimization. Applications of these functions have been discussed in the following section.

Table 12.4 Description of MATLAB function used for optimization

| Function | Description |
|------------------------------------|--|
| Curve fitting toolbox | |
| cftool | Open Curve Fitting Tool |
| fit | Fit model to data |
| sftool | Open Surface Fitting Tool |
| Optimization toolbox | |
| bintprog | Solve binary integer programming problems |
| fminbnd | Find minimum of single-variable function on fixed interval |
| fgoalattain | Solve multiobjective goal attainment problems |
| fmincon | Find minimum of constrained nonlinear multivariable function |
| fminimax | Solve minimax constraint problem |
| fminsearch | Find minimum of unconstrained multivariable function using derivative-free method |
| fminunc | Find minimum of unconstrained multivariable function |
| linprog | Solve linear programming problems |
| quadprog | Solve quadratic programming problems |
| Global Optimization problem | |
| ga | Find minimum of function using genetic algorithm |
| gamultiobj | Find minima of multiple functions using genetic algorithm |
| patternsearch | Find minimum of function using pattern search |
| simulannealbnd | Find unconstrained or bound-constrained minimum of function of several variables using simulated annealing algorithm |

Example 12.4

Write the MATLAB code for solving the problem given below

$$\text{Minimize } 31 - 11x_1 - 5x_2 + 3x_1^2 + x_2^2$$

Solution

The function “fminunc” is used for solving unconstrained multivariable optimization problem. Follow the steps given below

Step 1 Create an M file

```
function f = myfun(x)
f = 31-11*x(1)-5*x(2) + 3*x(1)^2 + x(2)^2;
```

Step 2 Save this file with file name “myfun.m”

Step 3 Write the code to find the optimized value near the point $x_0 = [1 \ 1]$

```
x0 = [1, 1];
[x, fval] = fminunc(@myfun, x0);
x, fval
```

Step 4 Run the programme

The result is

```
x =
    1.8333    2.5000
fval =
    14.6667
```

Example 12.5

Solve the optimization problem with constraints

Minimize $31 - 11x_1 - 5x_2 + 3x_1^2 + x_2^2$

subject to

$$x_1 + x_2 \leq 3$$

$$x_1^2 + x_2^2 = 8$$

Solution

Follow the similar steps as example 12.4

Step 1 Create a M file for the objective function (objfun.m):

```
function f = objfun(x)
f = 31-11*x(1)-5*x(2) + 3*x(1)^2 + x(2)^2;
```

Step 2 Create a M file for the constraint functions (confun.m):

```
function [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = x(1)+x(2)-3;
% Nonlinear equality constraints
ceq = x(1)^2+x(2)^2-8;
```

Step 3 Invoke constrained optimization routine:

```
x0 = [1, 1]; % Make a starting guess at the solution
```

```

options = optimset('Algorithm','active-set');
fmincon(@objfun,x0,[],[],[],[],[],[],@confun,options);
[x,fval]
which gives the result:
ans =
    2.8229    0.1771   23.0000

```

Example 12.6

Solve the problem in example 11.3. Use pattern search algorithm for the same.

Solution

The objective function is given by Eq. (11.21)

$$\begin{aligned}
 \text{Cr(VI) removal(\%)} = & 77.92 - 8.46(\text{pH}) + 9.41(\text{PAC}) + 3.86(\text{time}) \\
 & + 0.68(\text{pH})^2 - 3.53(\text{PAC})^2 - 7.33(\text{time})^2 \\
 & - 1.08(\text{pH})(\text{PAC}) + 1.08(\text{pH})(\text{time}) - 0.452(\text{time})(\text{PAC})
 \end{aligned} \tag{11.21}$$

We have to create a M file (.m) for the objective function

```

function f = crremoval(x)
f = -(77.92+9.41*x(1)+3.86*x(2)-3.53*x(1)^2-7.33*x(2)^2);

```

Save this file with file name “crremoval.m”

Write the given code in a separate M file:

```

x = patternsearch(@crremoval,x0)
x0 = [1 1];
[x,fval] = patternsearch(@crremoval,x0)

```

Run the file

```

which gives the result
x =
    1.3327    0.2630
fval =
   -84.6993

```

Note “patternsearch” function finds the minimum of any function. However, our problem is maximization problem. Therefore, we have written the objective function “crremoval” as the negative of the objective function (Eq. (11.21)).

Example 12.7

Solve the problem in example 11.3, using simulated annealing method.

Solution

MATLAB has inbuilt function “simulannealbnd” that can be used to solve unconstrained optimization problem.

Save the file with file name “crremoval”

```
function f = crremoval(x)
f = -(77.92+9.41*x(1)+3.86*x(2)-3.53*x(1)^2-7.33*x(2)^2);
```

Create another M file with code:

```
ObjectiveFunction = @crremoval;

X0 = [0.1 0.1]; % Starting point
[x,fval,exitFlag,output] = simulannealbnd(ObjectiveFunction,X0)
lb = [1 1];
ub = [10 10];

x =
    1.3329    0.2633
fval =
   -84.6993
```

Multiobjective optimization is possible in MATLAB using different methods. Here we will use genetic algorithm for solving MOO.

Example 12.8

Construct the Pareto front using genetic algorithm MOO technique.

Solution

Create the function with objective functions

```
function y = simple_multiobjective(x)
y(1) = 3+(x-4)^2;
y(2) = 16+(x-10)^2;
```

For plotting the objective functions use the code

```
% Plot two objective functions on the same axis
x = -10:0.5:10;
f1 = 3+(x-4).^2;
f2 = 16+(x-10).^2;
plot(x,f1);
```

```

hold on;
plot(x,f2,'r');
grid on;
title('Plot of objectives 'Objective 1' and 'Objective 2');

```

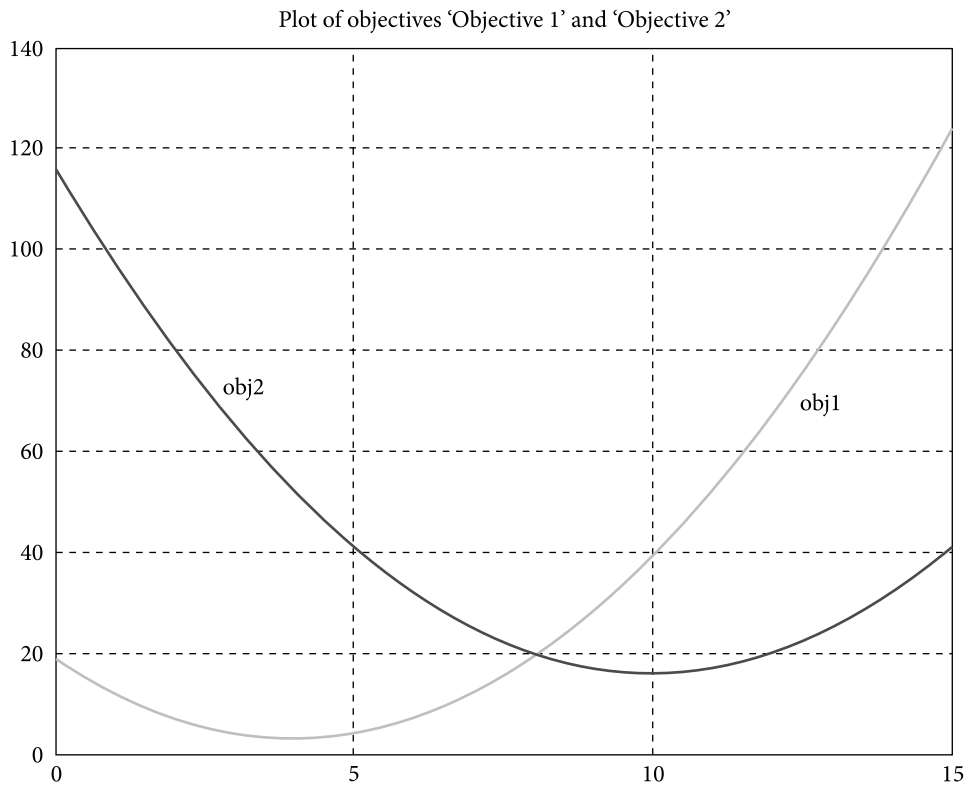


Fig. 12.8 Plot of objective 1 and objective 2

```

FitnessFunction = @simple_multiobjective;
numberOfVariables = 1;
[x,fval] = gamultiobj(FitnessFunction,numberOfVariables);
size(x)
size(fval)
A = []; b = [];
Aeq = []; beq = [];
lb = 1;
ub = 10;
x = gamultiobj(FitnessFunction,numberOfVariables,A,b,Aeq,beq,lb,ub);
options = gaoptimset('PlotFcns',{@gaplotpareto});
gamultiobj(FitnessFunction,numberOfVariables,[],[],[],[],lb,ub,options);

```

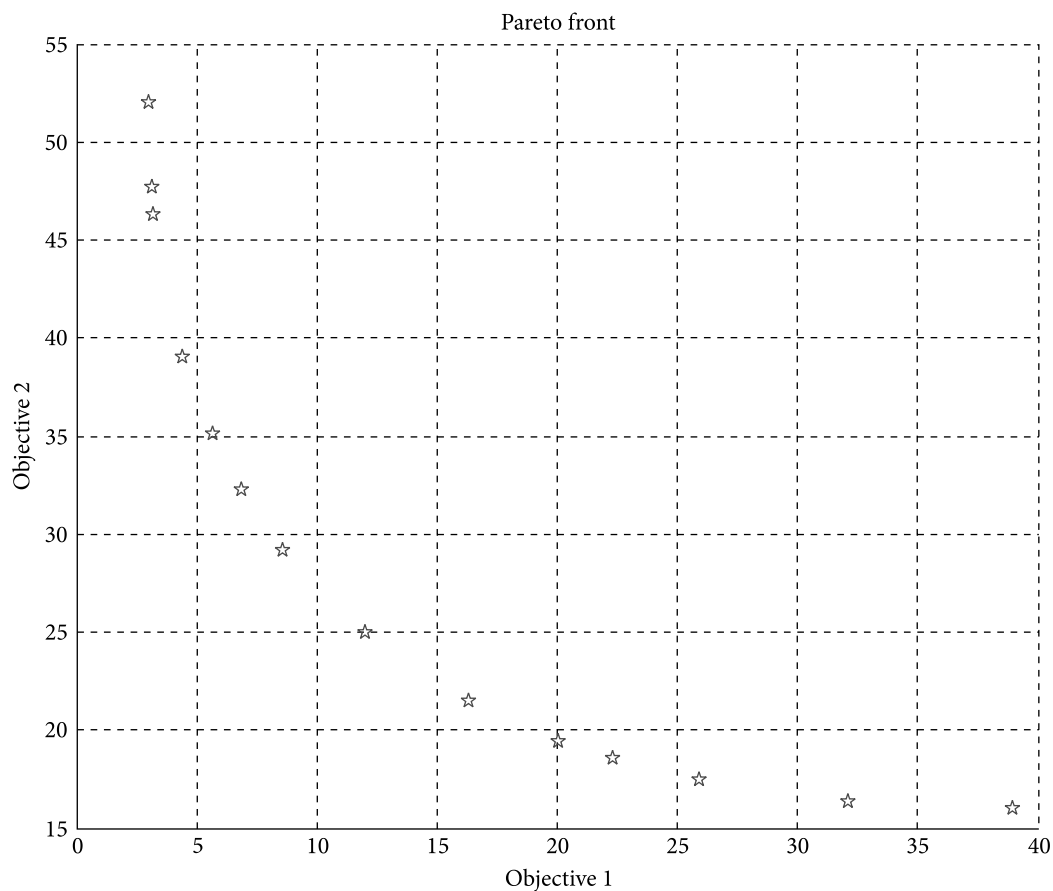


Fig. 12.9 Pareto front for obj1 and obj2

Examples 12.9

Consider the linear programming problem in example 6.2. Solve this problem using MATLAB.

Solution

The problem in example 6.2 is

$$\text{Maximize: } Z = 3x_1 + 2x_2$$

subject to constraints

$$2x_1 + x_2 \leq 10$$

$$x_1 + x_2 \leq 8$$

$$x_1 \leq 4$$

and

$$x_1 \geq 0, x_2 \geq 0$$

We need to convert the objective function as minimization problem. Therefore, the problem becomes

$$\text{Minimize: } F = -3x_1 - 2x_2$$

subject to constraints

$$2x_1 + x_2 \leq 10$$

$$x_1 + x_2 \leq 8$$

$$x_1 \leq 4$$

and

$$x_1 \geq 0, x_2 \geq 0$$

Write the follow code in a M file and run the file.

```
f = [-3; -2];
A = [2 1
     1 1
     1 0];
b = [10; 8; 4];
lb = zeros(2,1);
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);
x,fval
```

The output is as follows

Optimization terminated.

```
x =
    2.0000
    6.0000
fval =
   -18.0000
```

Note All materials are included with permission from MathWorks Inc.

12.3 MINITAB®

Minitab® is a statistical analysis software. It can be used for statistical research, design of experiment, and response surface methodology. We have already discussed about the theory of DOE and RSM

in chapter 11. In this section, we will learn how to use MINITAB® for developing an experimental run and optimize that process.

Getting started

The main window of MINITAB® looks like Fig. 12.10. Minitab Menu bar are two sub-windows; **Session** and **Worksheet1**. The **Session** window is where any non-graphical output is displayed (Fig. 12.16). Note that you can also type in commands in this window.

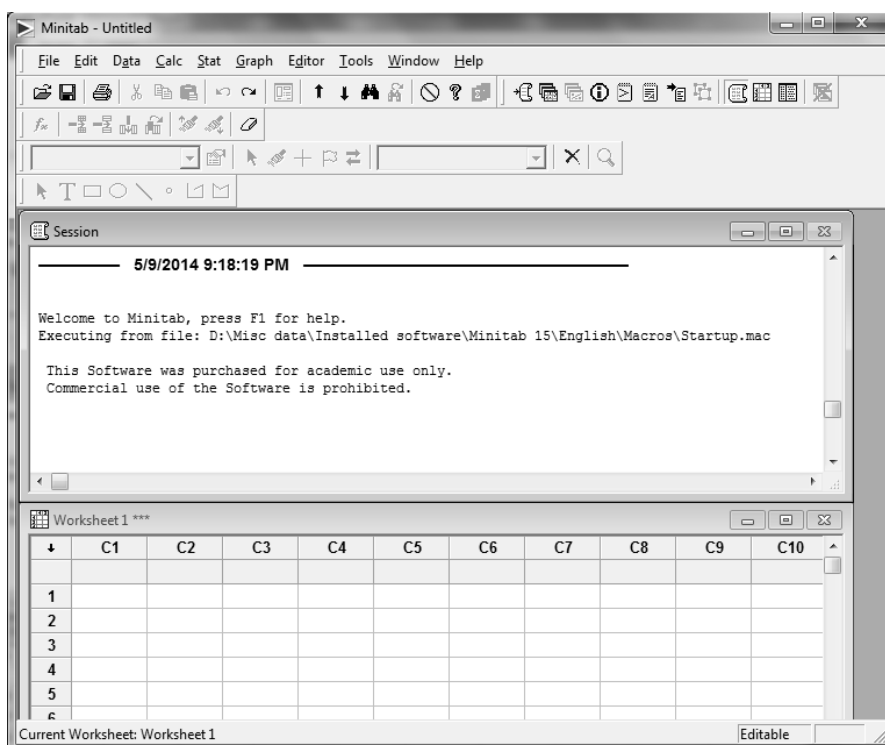


Fig. 12.10 MINITAB main window

“Worksheet” is the basic structural component of Minitab®. The **Worksheet1 ***** window is a spreadsheet, where we can type in and view our data.

The worksheet is a big rectangular array, or matrix, of cells organized into rows and columns as shown in Fig. 12.10. Each cell of this matrix contains one piece of data. This piece of data could be a numeric data, i.e. number; it could be a sequence of characters, such as a word or text data, i.e., an arbitrary string of numbers and letters. Most often data comes as numbers, for example 2.1, 5.8, ... but occasionally it appears in the form of a sequence of characters, such as white, green, male, female, etc. Typically, sequences of characters are used to identify the classifications for some variable of interest, e.g., gender, color. In Minitab®, the length of a piece of text data can be up to 80 characters. The extension .mtw indicate that this is a Minitab® worksheet.

Selection of DOE

To select the experimental design, follow the steps

Stat>>DOE>>Response Surface>>Create Response Surface Design
then select either Central Composite Design or Box–Behnken Design.

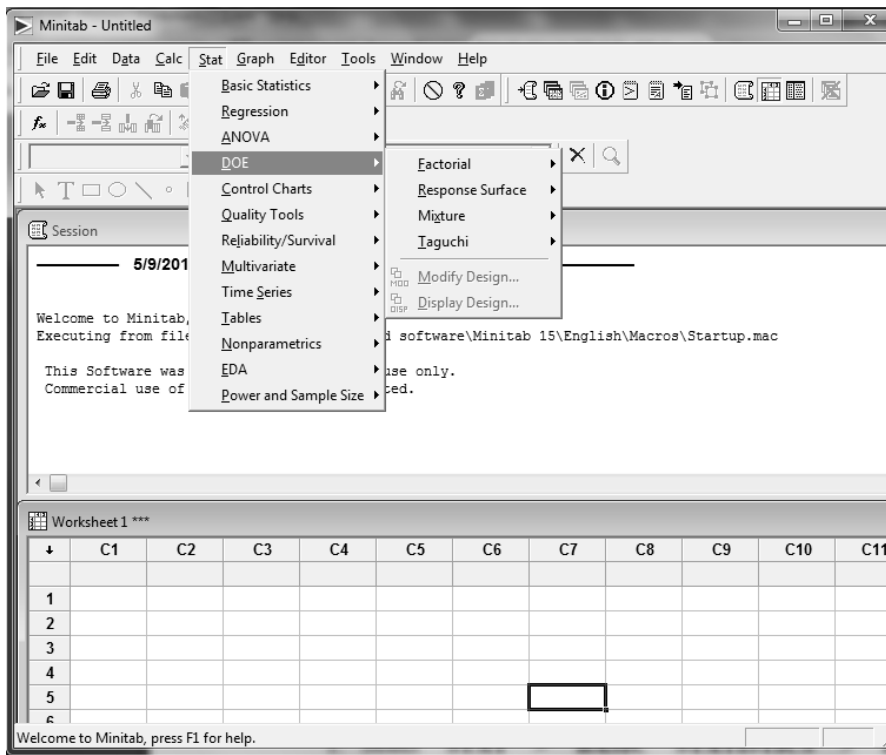


Fig. 12.11 MINITAB window for selecting DOE

Example 12.10

Consider the problem discussed by Dutta (2013) [Dutta (2013)]. Dye adsorption experiment was carried out with three different variables i.e., pH, TiO_2 dose, and contact time. Box–Behnken design was considered for this study. The maximum and minimum level of these variables are given in Table 12.5.

Table 12.5 Levels of independent variables for B–B design

| Independent variable | Symbol | Levels | | |
|---|--------|--------|-----|-------|
| | | –1 | 0 | +1 |
| pH | X_1 | 3.32 | 5 | 6.68 |
| TiO_2 dose (g.L^{-1}) | X_2 | 0.98 | 3.5 | 6.02 |
| Contact time (min) | X_3 | 2.45 | 26 | 49.55 |

Worksheet in Fig. 12.12 shows the experimental points for B–B design. Session sub-window shows that the total number of experiment required is 15.

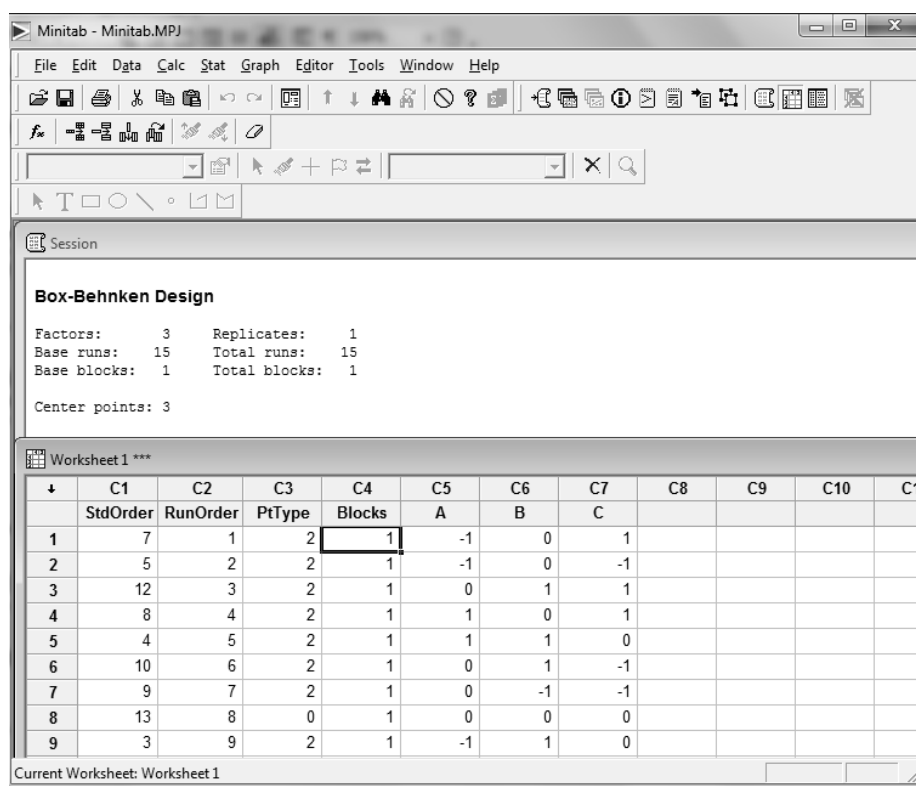


Fig. 12.12 MINITAB window with B–B design

Table 12.6 shows the experimental data for B–B design.

Table 12.6 Matrix of B–B design

| No of experiment | Coded value of independent variables | | | Response (% dye removal) |
|------------------|--------------------------------------|------------------|------|--------------------------|
| | pH | TiO ₂ | Time | |
| 1 | 1 | 0 | -1 | 2.4 |
| 2 | 0 | 0 | 0 | 58.4 |
| 3 | 1 | 1 | 0 | 33.1 |
| 4 | 0 | -1 | -1 | 7.18 |
| 5 | 0 | -1 | 1 | 16.58 |
| 6 | -1 | -1 | 0 | 60.85 |
| 7 | 0 | 0 | 0 | 58.4 |
| 8 | 1 | 0 | 1 | 17.15 |

| | | | | |
|----|----|----|----|-------|
| 9 | 0 | 1 | 1 | 67.78 |
| 10 | 0 | 1 | -1 | 37.74 |
| 11 | -1 | 0 | -1 | 66.14 |
| 12 | 1 | -1 | 0 | 2.1 |
| 13 | -1 | 0 | 1 | 90.81 |
| 14 | -1 | 1 | 0 | 85.9 |
| 15 | 0 | 0 | 0 | 58.4 |

Incorporate the Response (% removal) data from last column of Table 12.6 to the column 8 (C8) of the Worksheet1***. Then the window looks like Fig. 12.13.

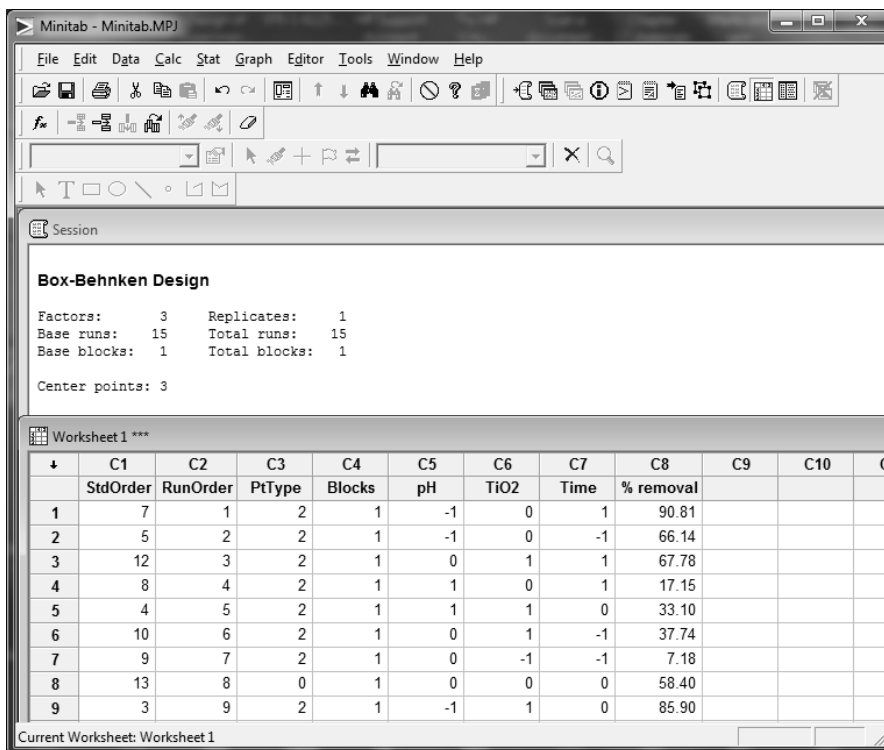


Fig. 12.13 MINITAB window with data for B-B design

After data input, analysis of the response surface design is required. Follow the given steps for this purpose.

Stat>>DOE>>Response Surface>>Analyze Response Surface Design

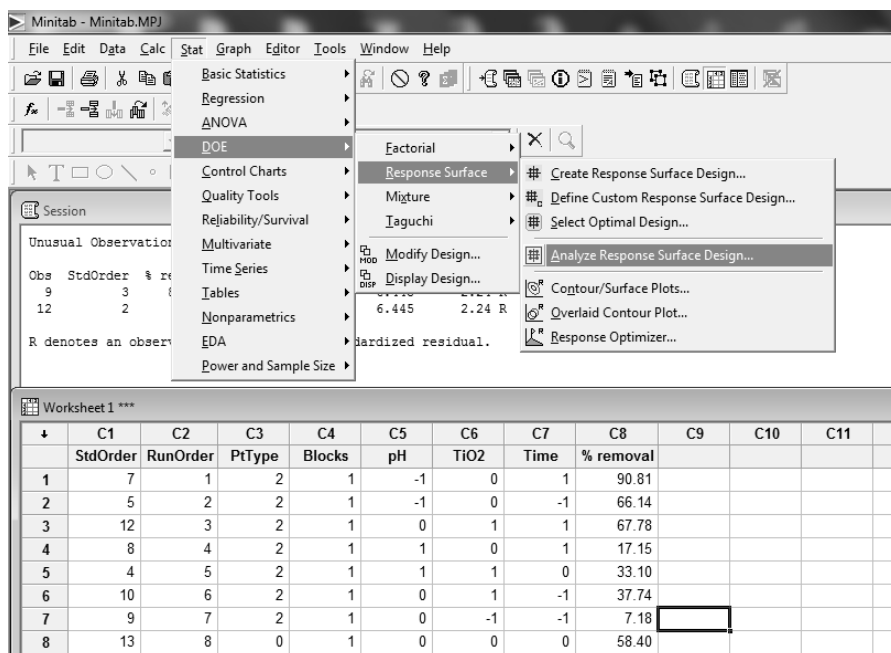


Fig. 12.14 MINITAB window for analyzing the response surface design

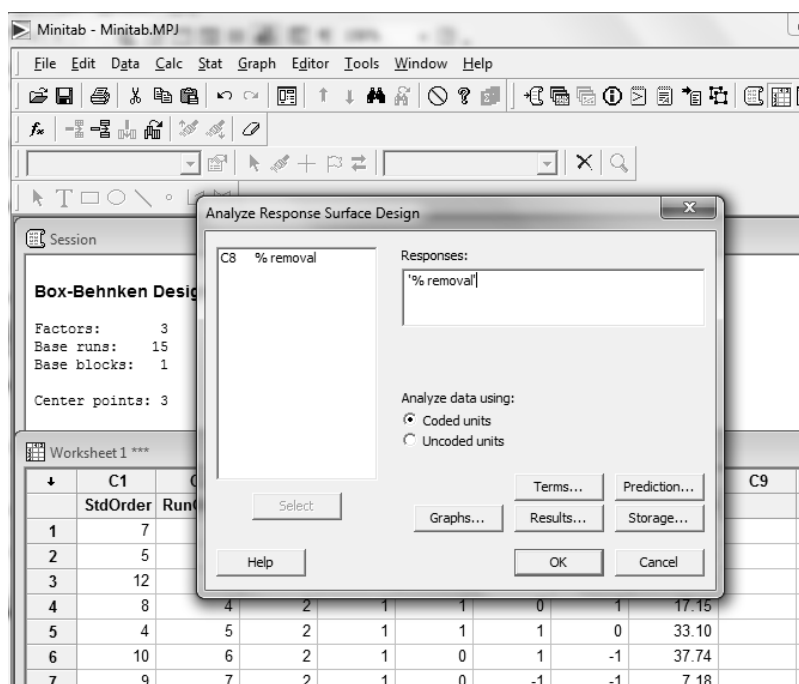


Fig. 12.15 MINITAB window for selecting “response”

A sub-window will appear as shown in Fig. 12.15, we have to select C8 %removal as “response”. Then select options for result, graph, storage etc. Press OK button to get result.

The calculated values of the coefficients are represented in Fig. 12.16. Other results will appear in “session” sub-window. Table 12.7 shows the results of ANOVA study.

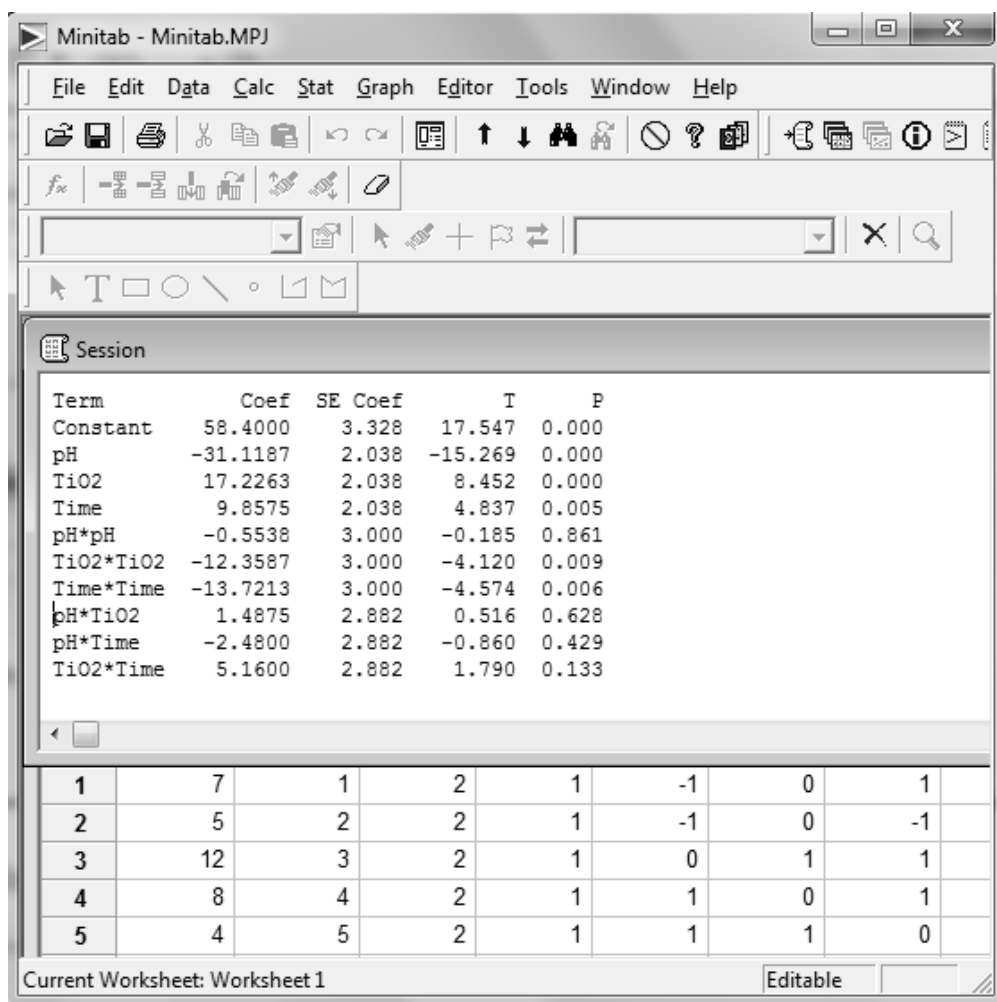


Fig. 12.16 MINITAB window with results

Table 12.7 ANOVA for percentage dye removal

| Source | DF | Seq SS | Adj SS | Adj MS | F | P |
|------------|----|---------|----------|---------|--------|-------|
| Regression | 9 | 12215.9 | 122.15.9 | 1357.33 | 40.85 | 0.000 |
| Linear | 3 | 10900.3 | 10900.3 | 3633.42 | 109.34 | 0.000 |
| Square | 3 | 1175.9 | 1175.9 | 391.98 | 11.80 | 0.010 |

| | | | | | | |
|----------------|----|---------|-------|-------|------|-------|
| Interaction | 3 | 139.7 | 139.7 | 46.58 | 1.40 | 0.345 |
| Residual error | 5 | 166.2 | 166.2 | 33.23 | | |
| Lack-of-fit | 3 | 166.2 | 166.2 | 55.38 | | |
| Pure error | 2 | 0.0 | 0.0 | 0.0 | | |
| Total | 14 | 12382.1 | | | | |

The final equation is

$$\begin{aligned}
 \text{Dye removal (\%)} = & 58.40 - 31.119(\text{pH}) + 17.226(\text{TiO}_2) + 9.858(\text{Time}) \\
 & - 0.554(\text{pH})^2 - 12.359(\text{TiO}_2)^2 - 13.721(\text{Time})^2 \\
 & + 1.488(\text{pH})(\text{TiO}_2) - 2.48(\text{pH})(\text{Time}) + 5.16(\text{TiO}_2)(\text{Time})
 \end{aligned} \quad (12.2)$$

To construct response surfaces and contour plots open the following link

Stat>>DOE>>Response Surface>>Contour/Surface Plots

Figures 12.17a, 12.17b, and 12.17c, represent the response surfaces for the response Eq. (12.2). The corresponding contours are given in 12.18a, 12.18b, 12.18c,

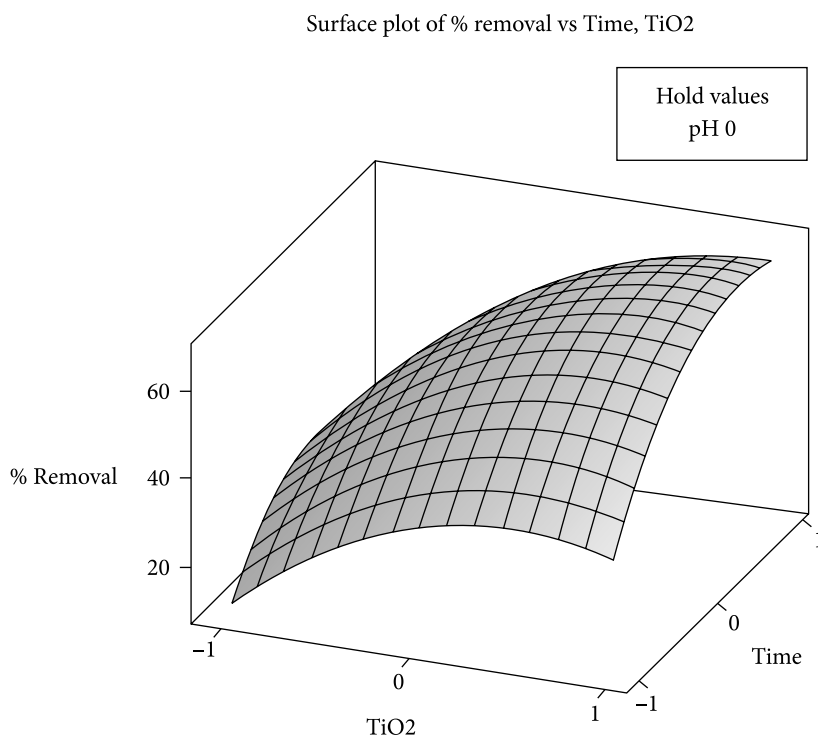


Fig. 12.17a Response surface of Eq. (12.2) at pH = 0

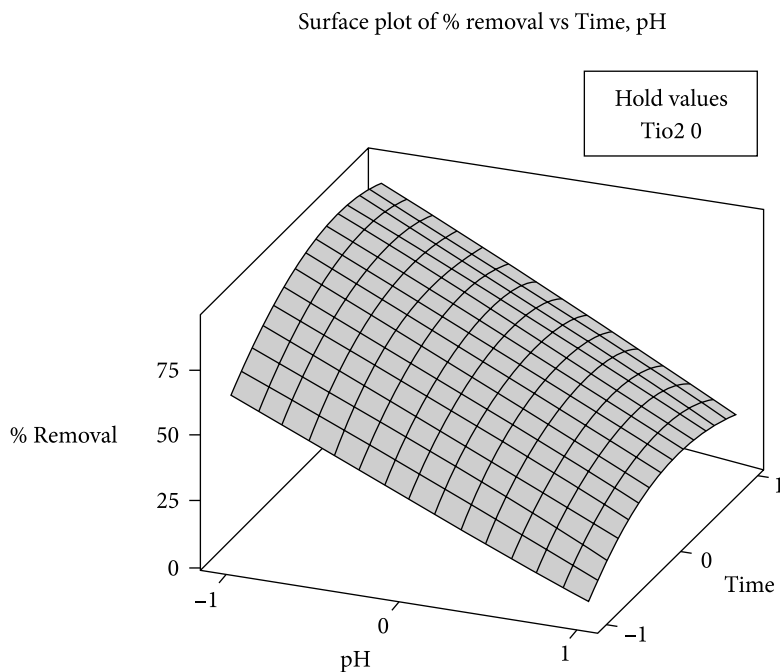


Fig. 12.17b Response surface of Eq. (12.2) at $\text{TiO}_2 = 0$

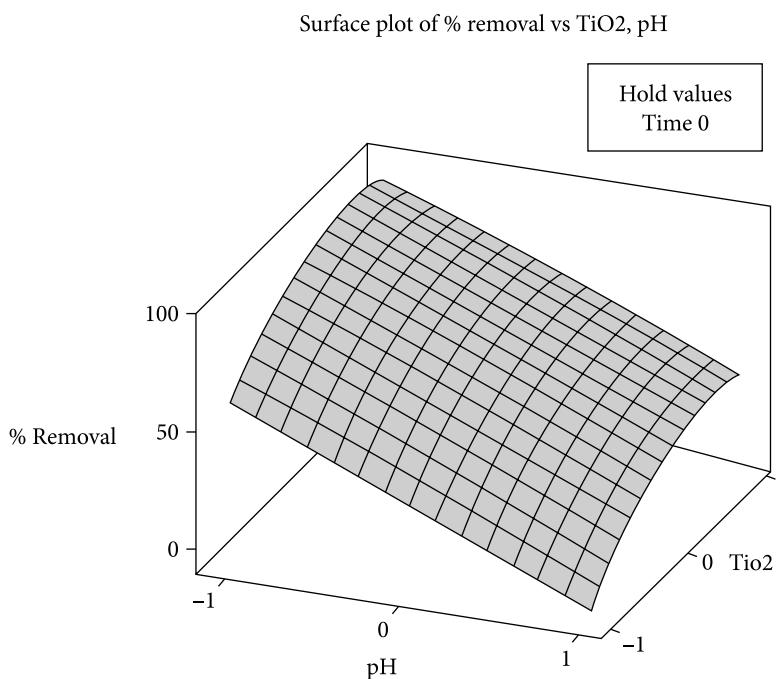


Fig. 12.17c Response surface of Eq. (12.2) at $\text{Time} = 0$

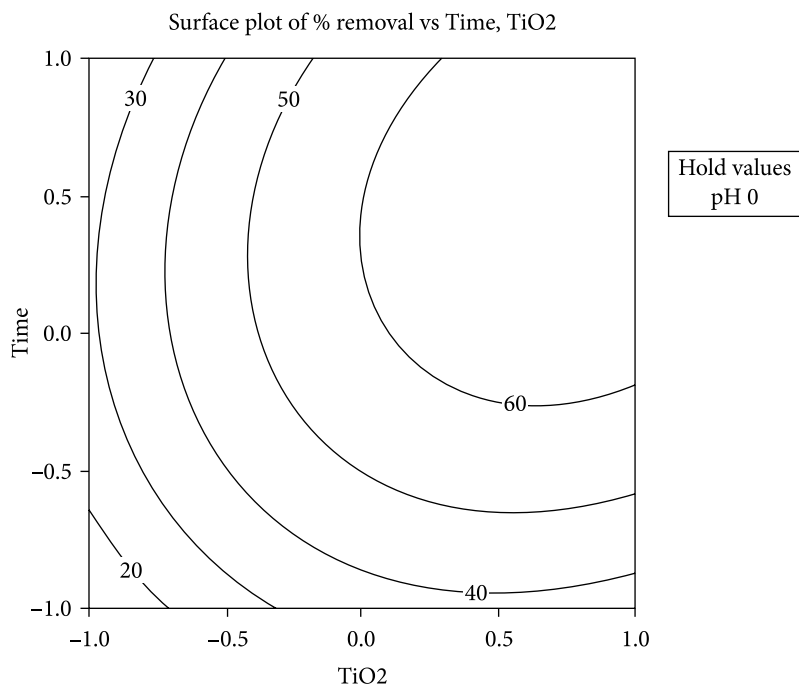


Fig. 12.18a Contour of the response Eq. (12.2) at pH = 0

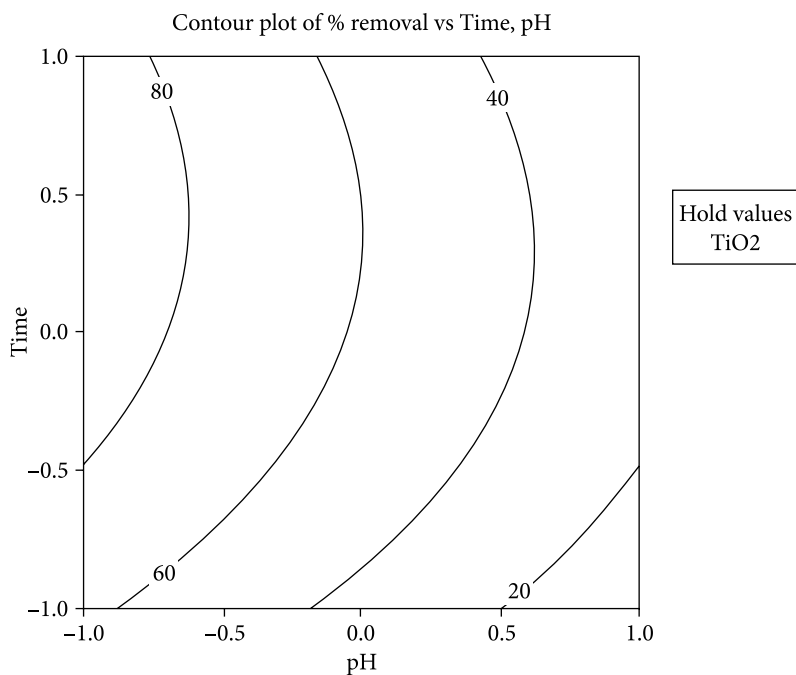


Fig. 12.18b Contour of the response Eq. (12.2) at TiO₂ = 0

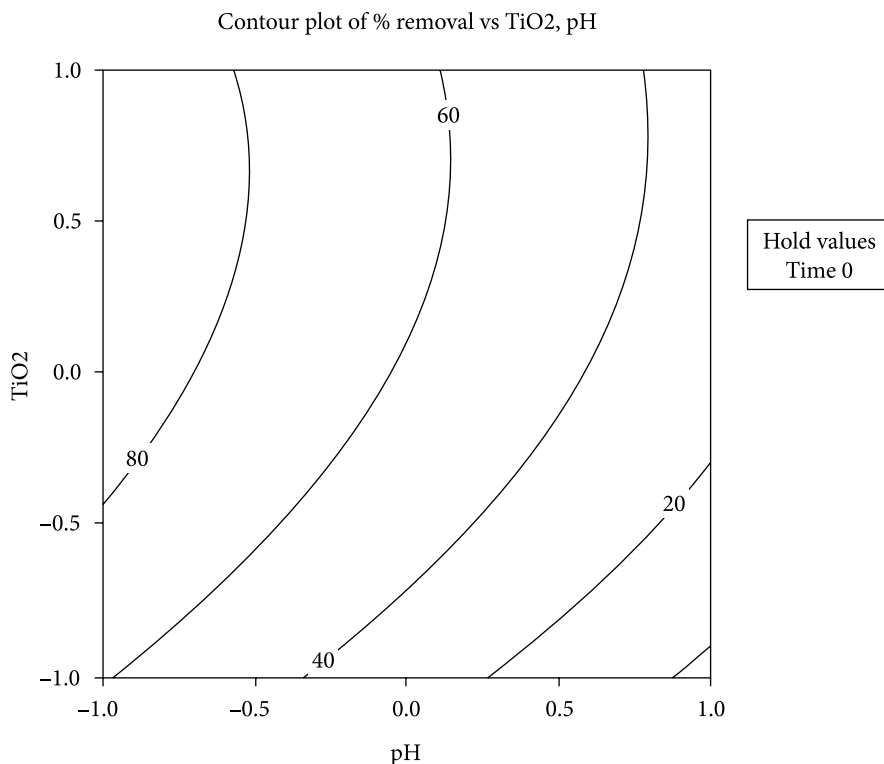


Fig. 12.18c Contour of the response Eq. (12.2) at Time = 0

The optimum values of pH, TiO₂ dose, and time are 3.32, 5.4 g.L⁻¹, and 40.0 min, respectively. Dye removal at this optimum condition is 98.61%.

Note Portions of information contained in this publication/book are printed with permission of Minitab Inc. All such material remains the exclusive property and copyright of Minitab Inc. All rights reserved.

12.4 GAMS

The General Algebraic Modeling System (GAMS) is a high-level modeling system, which can be used for mathematical programming and optimization. It is composed of a language compiler and a stable of integrated high-performance solvers. GAMS is adapted for complex, large scale modeling applications, and it allow us to build large maintainable models that can be modified quickly to new model depending on the situation.

Input/output

The main window of GAMS looks like Fig. 12.19.

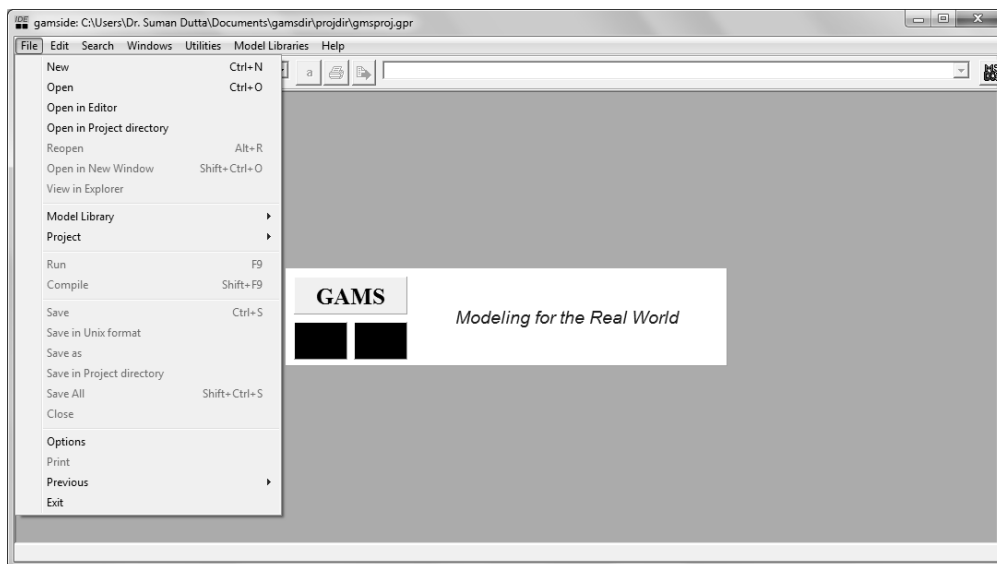


Fig. 12.19 Main window of GAMS

Open a new file, the input model file will be saved as “.gms” format: filename.gms

When input file is ready, the model is solved using a solution procedure given in Table 12.10 and the out with error message sent to corresponding “.lst” file.

GAMS Statement Formats

The following syntax has been used to write any GAMS command:

- Line that starts with an * (asterisk) as first character is comment, see the example 12.11 and 12.12.
- Line with a dollar sign (\$) in the very first character is a GAMS input/output option. For example, \$ include “filename” copies in the content of “filename” as if it had been typed at this point. There is no end punctuation for such \$-option lines.
- All other statements in GAMS are written over one or more lines and end with a semicolon; commas are used to separate the component items.
- For better readability, extra spaces, line breaks, and blank lines may be added without any effect.
- GAMS is not case sensitive. Therefore, AA, aa and Aa have the same meaning.
- Declaration statements such as free variable(s) and equations(s) may have some words of double-quoted “explanatory text” (i.e., “total profit” in Example 12.12) explaining the meaning of each defined item immediately after its name is specified. This explanatory text will come with the item on outputs to make results easier to understand.
- Names in GAMS are made of up to 9 letters and digits, beginning with a letter. Spaces, internal commas and other special characters are not allowed, but underlines () may be used.

- All GAMS command words (e.g., model, variable, table, equation) and function names (e.g., sum, sin, log) are reserved, and should not be used for declared names. Giving name to the entities with some standard computer words such as if, else, elseif, for, while, file, and system also causes errors.
- Subscripts on variables and constraints are enclosed in parentheses whereas explicit (non-varying) subscripts are enclosed in quotes.
- Numerical constants in statements may have a decimal point, but they should not include commas (i.e., 20000 is correct and 20,000 is incorrect).

Defining Variables

Variables can be define in different forms and the details are given in Table 12.8.

Table 12.8 Different variable type and their GAMS keyword

| GAMS keyword | Variable type |
|----------------------|---------------------------------------|
| free variable(s) | unrestricted (continuous) variable(s) |
| positive variable(s) | nonnegative (continuous) variable(s) |
| negative variable(s) | nonpositive (continuous) variable(s) |
| binary variable(s) | 0–1 variable(s) |
| integer variable(s) | nonnegative integer variable(s) |

The following relational operators are used in GAMS

Table 12.9 Different relational operator in GAMS

| Relation | GAMS syntax |
|-------------------------------------|-------------|
| Equality (=) | = e = |
| Less than or equal to (\leq) | = l = |
| Greater than or equal to (\geq) | = g = |

Table 12.10 Available solution procedure in GAMS

| Solution procedure | Description |
|--------------------|---|
| lp | Used for linear programming |
| nlp | Used for nonlinear programming |
| qcp | Used for quadratic constraint programming |
| dnlp | Used for nonlinear programming with discontinuous derivatives |
| mip | Used for mixed integer programming |
| rmip | Used for relaxed mixed integer programming |
| miqcp | Used for mixed integer quadratic constraint programming |

| | |
|--------|---|
| minlp | Used for mixed integer nonlinear programming |
| rmicqp | Used for relaxed mixed integer quadratic constraint programming |
| rminlp | Used for relaxed mixed integer nonlinear programming |
| mcp | Used for mixed complementarity problems |
| mpec | Used for mathematical programs with equilibrium constraints |
| cns | Used for constrained nonlinear systems |

Example 12.11 is a very simple problem, which optimize an unconstrained optimization problem.

Example 12.11

Simple unconstrained optimization problem:

Solution

```
* Unconstrained nonlinear programming
free variable x1, x2;
free variable objective_function;
equations
obj "objective_function";
obj..
31-11*x1-5*x2+3*x1**2+x2**2 =e= objective_function;
model obj_fun /all/;
solve obj_fun using nlp minimizing objective_function;
```

Example 12.12

The daily profit of a chemical company is given in Example 6.1. Write the code for solving the problem. (Use linear programming method)

Solution:

```
* S. Dutta Example 6.1 file profit.gms
* This code is written for the book "Optimization in Chemical
Engineering"
free variable
daily_profit "total profit";
positive variables
x1 "production rate 1",
x2 "production rate 2";
equations
obj "max total profit",
supplyA "supplyA",
supplyB "supplyB";
obj..
14*x1 + 11*x2 =e= daily_profit;
```

```

supplyA..
3*x1 + 2*x2 =l= 36;
supplyB..
x1 + x2 =l= 14;
model profit /all/;
solve profit using lp maximizing daily_profit;

```

Example 12.13

Consider the problem given in example 6.6, use integer programming method for solving the problem.

Solution

```

* S. Dutta Example 6.6 file sdutta.gms
option optcr=0.0;
free variable objective_function;
integer variables
x1, x2;
equations
obj "maximize objective function",
constraint_1,
constraint_2;
obj..
5*x1 + 7*x2 =e= objective_function;
constraint_1..
x1 + x2 =l= 6;
constraint_2..
5*x1 + 9*x2 =l= 43;
model untitled_8 /all/;
solve untitled_8 using mip maximizing objective_function;
Display x1.l, x2.l;

```

Example 12.14

Write the GAMS code for the warehouse problem given in example 12.2.

Solution

```

* Code written by S. Dutta for the book Optimization in
Chemical Engineering
Sets
i company warehouse / ww1, ww2, ww3, ww4, ww5 /
j vendors / v1, v2, v3, v4, v5, v6 / ;
Parameters
a(i) capacity of warehouse
/ ww1 60, ww2 55, ww3 51, ww4 41, ww5 52 /

```

```

b(j) vendor widget demand
/ v1 35, v2 37, v3 22, v4 43, v5 38, v6 32 / ;
Table c(i,j) shipping cost per widget
      v1  v2  v3  v4  v5  v6
WW1    6   2   7   4   2   9
WW2     4   9   5   3   8   6
WW3    7   6   7   3   9   2
WW4    2   3   9   7   5   3
WW5    5   6   2   3   8   1;

Variables
x(i,j) supply of widget from warehouse i to vendor j
z total transportation cost ;
Positive Variable x ;
Equations
cost define objective function
supply(i) supply limit at warehouse i
demand(j) demand of vendor j ;
cost ..
z =e= sum((i,j), c(i,j)*x(i,j));
supply(i) ..
sum(j, x(i,j)) =l= a(i);
demand(j) ..
sum(i, x(i,j)) =g= b(j);
Model transport /all/ ;
Solve transport using lp minimizing z;
Display x.l, x.m ;

```

Example 12.15

This example elucidates blending in a petroleum refinery.

A refinery produces three grades of gasoline (e.g. regular, premium, and low lead) from four different petroleum stocks. The availability and cost of these petroleum stocks are in Table 12.11.

Table 12.11 Availability and cost of petroleum stocks

| Petroleum stock | Availability (barrels/day) | Cost (\$/barrel) |
|-----------------|-------------------------------|---------------------|
| 1 | 5500 | 9 |
| 2 | 2500 | 7 |
| 3 | 4000 | 13 |
| 4 | 1500 | 6 |

The problem is to determine the optimal usage of the four petroleum stocks that will maximize the profit.

The specification and selling price of each grade is given in Table 12.12.

Table 12.12 Specification and selling price

| Grade | Specification | Selling price (\$/barrel) |
|--------------|---|---------------------------|
| regular (R) | (1) not less than 40% of S1 (2) not more than 20% of S2 (3) not less than 30% of S3 | 12.00 |
| premium (P) | (4) not less than 40% of S3 | 18.00 |
| low lead (L) | (5) not more than 50% of S2 (6) not less than 10% of S1 | 10.00 |

Solution

Consider, x_{ij} is barrel of petroleum stock i used in grade j per day ($i = 1, 2, 3, 4$ and $j = R, P, L$)

Objective function:

$$\begin{aligned}
 \text{Maximize } P &= 12(x_{1R} + x_{2R} + x_{3R} + x_{4R}) + 18(x_{1P} + x_{2P} + x_{3P} + x_{4P}) \\
 &\quad + 10(x_{1L} + x_{2L} + x_{3L} + x_{4L}) - 9(x_{1R} + x_{1P} + x_{1L}) - 7(x_{2R} + x_{2P} + x_{2L}) \\
 &\quad - 13(x_{3R} + x_{3P} + x_{3L}) - 6(x_{4R} + x_{4P} + x_{4L}) \\
 &= 3x_{1R} + 5x_{2R} - x_{3R} + 6x_{4R} + 9x_{1P} + 11x_{2P} + 5x_{3P} + 12x_{4P} + x_{1L} + 3x_{2L} - 3x_{3L} + 4x_{4L} \quad (12.3)
 \end{aligned}$$

Constraints:

Availability:

$$x_{1R} + x_{1P} + x_{1L} \leq 5000 \quad (12.4a)$$

$$x_{2R} + x_{2P} + x_{2L} \leq 2400 \quad (12.4b)$$

$$x_{3R} + x_{3P} + x_{3L} \leq 4000 \quad (12.4c)$$

$$x_{4R} + x_{4P} + x_{4L} \leq 1500 \quad (12.4d)$$

Specification for regular (R) grade

$$1. \quad x_{1R} / (x_{1R} + x_{2R} + x_{3R} + x_{4R}) \geq 0.40$$

$$\text{or } 0.6x_{1R} - 0.4x_{2R} - 0.4x_{3R} - 0.4x_{4R} \geq 0 \quad (12.5a)$$

$$2. \quad x_{2R} / (x_{1R} + x_{2R} + x_{3R} + x_{4R}) \leq 0.20$$

$$-0.2x_{1R} + 0.8x_{2R} - 0.2x_{3R} - 0.2x_{4R} \leq 0 \quad (12.5b)$$

$$3. \quad x_{3R} / (x_{1R} + x_{2R} + x_{3R} + x_{4R}) \geq 0.30$$

$$-0.3x_{1R} - 0.3x_{2R} + 0.7x_{3R} - 0.3x_{4R} \geq 0 \quad (12.5c)$$

Specification for premium (P) grade

$$4. \quad x_{3P} / (x_{1P} + x_{2P} + x_{3P} + x_{4P}) \geq 0.40$$

$$-0.4x_{1P} - 0.4x_{2P} + 0.6x_{3P} - 0.4x_{4P} \geq 0 \quad (12.5d)$$

Specification for low lead (L) grade

$$5. \quad x_{3L} / (x_{1L} + x_{2L} + x_{3L} + x_{4L}) \leq 0.50$$

$$-0.5x_{1L} + 0.5x_{2L} - 0.5x_{3L} - 0.5x_{4L} \leq 0 \quad (12.5e)$$

$$6. \quad x_{1L} / (x_{1L} + x_{2L} + x_{3L} + x_{4L}) \geq 0.10$$

$$0.9x_{1L} - 0.1x_{2L} - 0.1x_{3L} - 0.1x_{4L} \geq 0 \quad (12.5f)$$

and

$$x_{ij} \geq 0 \quad (12.6)$$

GAMS code

```
* S.Dutta Example 12.15 file blending.gms
*This code is written for the book Optimization in Chemical
Engineering
free variable profit "profit";
positive variables x1R,x2R,x3R,x4R,x1P,x2P,
x3P,x4P,x1L,x2L,x3L,x4L;
equations
*Objective function
```

```

obj "max profit",
*Availability constraints
avil1 "availability of stock 1",
avil2 "availability of stock 2",
avil3 "availability of stock 3",
avil4 "availability of stock 4",

*Specification constraints
spec1 "specification_1 for R",
spec2 "specification_2 for R",
spec3 "specification_3 for R",
spec4 "specification_4 for P",
spec5 "specification_5 for L",
spec6 "specification_1 for L";
obj..
3*x1R+5*x2R-x3R+6*x4R+9*x1P+11*x2P+5*x3P
+12*x4P+x1L+3*x2L-3*x3L+4*x4L =e= profit;
avil1..
x1R+x1P+x1L =l= 5500;
avil2..
x2R+x2P+x2L =l= 2500;
avil3..
x3R+x3P+x3L =l= 4000;
avil4..
x4R+x4P+x4L =l= 1500;
spec1..
0.6*x1R-0.4*x2R-0.4*x3R-0.4*x4R =g= 0;
spec2..
-0.2*x1R+0.8*x2R-0.2*x3R-0.2*x4R =l= 0;
spec3..
-0.3*x1R-0.3*x2R+0.7*x3R-0.3*x4R =g= 0;
spec4..
-0.4*x1R-0.4*x2R+0.6*x3R-0.4*x4R =g= 0;
spec5..
-0.5*x1R+0.5*x2R-0.5*x3R-0.5*x4R =l= 0;
spec6..
0.9*x1R-0.1*x2R-0.1*x3R-0.1*x4R =g= 0;
model blending /all/;
solve blending using lp maximizing profit;

```

Note All materials are incorporated with permission from GAMS Software GmbH

Summary

- The available commercial software/software tools are very useful for solving complicated optimization problems. LINGO, MATLAB, MINITAB®, and GAMS are discussed in this chapter. The optimization problems can be written easily using proper syntax. Different types of optimization problems like LP, NLP, MINLP have been solved using these software. Chemical engineering problems such as blending of petroleum product has been solved using GAMS.

Review Questions

- 12.1 Write the code in LINGO for solving the blending problem given in example 12.15.
- 12.2 What information we get from the LINGO solver status window?
- 12.3 Can we use simulated annealing algorithm to solve multi-objective optimization problem?
- 12.4 Use the given data to fit a linear equation. Write the code in MATLAB.

| | | | | | | |
|---|---|-----|-----|-----|-----|-----|
| x | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 |
| y | 0 | 1.2 | 3.3 | 4.8 | 5.9 | 7.2 |

- 12.5 How do you find the optimum point from the contour developed by RSM ?
- 12.6 Construct an outline for experimental design using factorial design.
- 12.7 Find the error in the given GAMS code
 $z = 20x_1 + 13x_2$;
 write the correct code.
- 12.8 Write the GAMS code for finding the optimum of the Eq. (12.2) with additional constraints
 $3 \leq \text{pH} \leq 7$
 $0 \leq \text{TiO}_2 \leq 10$
 $0 \leq \text{time} \leq 50$

References

Dutta S. *Optimization of Reactive Black 5 Removal by Adsorption Process using Box–Behnken Design*, 'Desalination and Water Treatment', 51(2013): 7631–38.

www.gams.com

www.lindo.com

www.mathworks.in

www.minitab.com